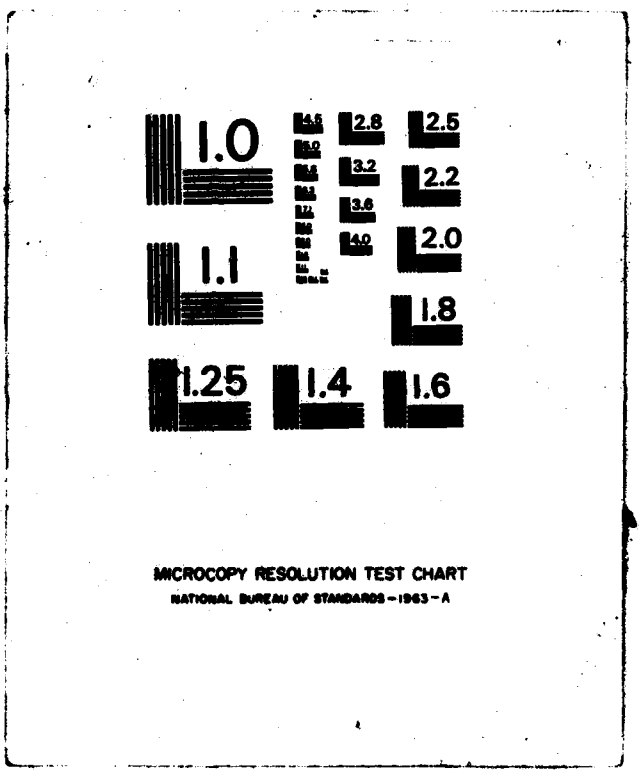
AD-A120 902 A FAST ALGORITHM THAT MAKES MATRICES OPTIMALLY SPARSE 1/1
(U) STANFORD UNIV CA SYSTEMS OPTIMIZATION LAB
A J HOFFMAN ET AL. SEP 82 SOL-82-13 ARO-16470.17-MA
UNCLASSIFIED N00014-75-C-0267 F/G 12/1 NL

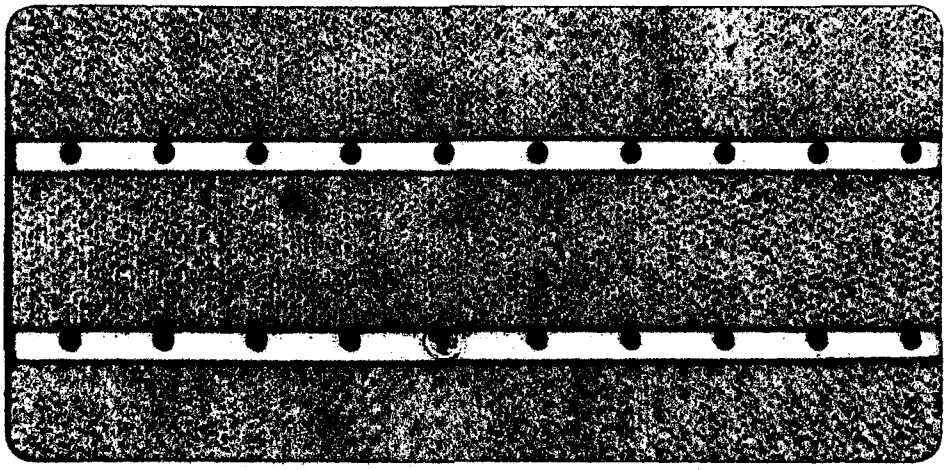

1.0
1.1
1.25
1.4
1.6
2.8
3.2
3.6
4.0
2.5
2.2
2.0
1.8
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ARO 16470.17-MA

(12)

Systems
Optimization
Laboratory









Department of Operations Research
Stanford University
Stanford, CA 94305

SYSTEMS OPTIMIZATION LABORATORY
DEPARTMENT OF OPERATIONS RESEARCH
STANFORD UNIVERSITY
STANFORD, CALIFORNIA 94305

# A Fast Algorithm That Makes Matrices Optimally Sparse†

by

Alan J. Hoffman *
S. Thomas McCormick **

TECHNICAL REPORT SOL 82-13
September, 1982

†Presented at the Silver Jubilee Conference in Combinatories University of Waterloo, Waterloo, Ontario, Canada, June 14-July 2, 1982.

* IBM Thomas J. Watson Center, Yorktown Heights, New York 10598.
New York 10598
**Department of Operations Research, Stanford Univeristy, Stanford, California 94305.

Research and reproduction of this report were partially supported by the Department of Energy Contract AM03-76SF00326, PA# DE-AT03-76ER72018; Office of Naval Research Contract N00014-75-C-0267; National Science Foundation Grants MCS-8119774, MCS-7926009 and ECS-8012974; Army Research Office Contract DAA29-79-C-0110.

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do NOT necessarily reflect the views of the above sponsors.

# A FAST ALGORITHM THAT MAKES MATRICES OPTIMALLY SPARSE

by

Alan J. Hoffman

IBM Thomas J. Watson Research Center
Yorktown Heights, New York 10598

and

S. Thomas McCormick

Department of Operations Research
Stanford University
Stanford, California 94305

May 1982

## ABSTRACT

Under a non-degeneracy assumption on the non-zero entries of a given sparse matrix, a polynomially-bounded algorithm is presented that performs row operations on the given matrix which reduce it to a sparsest possible matrix with the same row space. For each row of the matrix, the algorithm performs a maximum cardinality matching on the bipartite graph associated with a submatrix which is induced by that row. The dual of the optimal matching then specifies the row operations that will be performed on that row. ~~We also describe~~ a variant algorithm that processes the matrix in place, thus conserving storage and time. The modifications needed to apply the algorithm to matrices that do not necessarily satisfy the non-degeneracy assumption are also described. A particularly promising application of this algorithm is in reduction of linear constraint matrices.

running head: Making Matrices Optimally Sparse

The research of the second author was partially supported by the Department of Energy Contract AM03-76SF00326, PA# DE-AT03-76ER72018; Army Research Office Contract DAA29-79-C-0110; Office of Naval Research Contract N00014-74-C-0267; National Science Foundation Grants MCS76-81259, MCS-79260099 and ECS-8012974.

## 1. Introduction

An important factor in our present ability to solve many large-scale numerical problems is the recognition that these problems are nearly always sparse, and that taking advantage of sparsity can turn a hitherto practically unsolvable problem into a solvable one. Perhaps the best example of this is in large-scale linear programming, where highly refined sparse matrix factorization routines have allowed problems with huge coefficient matrices to be solved (see *e.g.*, Duff (1980) or Bunch and Rose (1976)). However, although sparsity is known to be helpful, relatively little attention seems to have been paid to techniques that economically *increase* sparsity (decrease density), thereby improving the efficiency of sparse algorithms. In this context, this paper considers the Sparseness Problem (SP):

Given a large, sparse system of linear equations

$$Ax = b, \tag{1}$$

find an equivalent system

$$\overline{A}x = \overline{b} \tag{2}$$

which has the minimum possible number of non-zero entries in $\overline{A}$.

Constraints of the form (1) are among the most common in large-scale optimisation, so that it is potentially very useful to solve (SP). Under a non-degeneracy assumption, we shall present an efficient algorithm that solves (SP) using maximum cardinality matching. Sections 2–4 will assume familiarity with notions of graph theory and maximum cardinality bipartite matching (see, *e.g.*, Lawler (1976)). Section 2 develops most of the machinery needed for the proof, and uses it to derive an algorithm that solves a subproblem of (SP). In Section 3 we use the algorithm of Section 2 to construct the full algorithm, and prove that it solves (SP). We then give a variant algorithm that uses less space and show that it also solves (SP). Section 4 discusses the modifications necessary to apply the algorithm on matrices that do not necessarily satisfy the non-degeneracy assumption. Finally, Section 5 considers further questions raised by this research.

## 2. Transforms and the One Row Algorithm

In this paper we shall assume that the matrix $A$ in (1) has full row rank. We know from linear algebra that (2) is equivalent to (1) if and only if $\overline{A} = TA$ and $\overline{b} = Tb$ for some square non-singular matrix $T$. We are aiming for a general algorithm that makes no assumptions about any special structure in $A$, and thus can find $T$ almost solely from the *sparsity pattern* of $A$ (the positions of the non-zeros in $A$). What can go wrong in this aim is that we can encounter

"unexpected" cancellation. To illustrate, consider the following two $A$'s, with the same sparsity pattern, treated with the same $T$:

$$TA_1 = \begin{pmatrix} 1 & -1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix};$$

$$TA_2 = \begin{pmatrix} 1 & -1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 2 & 3 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -1 & -2 \\ 0 & 1 & 1 & 2 & 3 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}.$$

In both cases $T$ represents the unique linear transformation that adds the multiples of rows 2 and 3 to row 1 which makes $a_{12}$ zero and avoids fill-in in $a_{13}$. In the first case the sparsity increased, in the second case it decreased. The difficulty is that $A_1$ has some dependent submatrices that are not apparent from the sparsity pattern alone. The possibility of this phenomonon makes solving (SP) too difficult in general, as shown by the following result.

**Theorem 1.** (Stockmeyer (1982)) (SP) is NP-Hard in general. (See the Appendix for the proof.) □

Thus, to get a polynomial algorithm for (SP), we must make some assumption about $A$. Suppose that $A$ is $m \times n$, and let $R \subseteq \{1,\ldots,m\}$, $C \subseteq \{1,\ldots,n\}$. We denote the submatrix of $A$ indexed by the rows in $R$ and the columns in $C$ by $A_{RC}$. The sparsity pattern of $A_{RC}$ naturally induces a bipartite graph $\mathcal{G}_{RC} = (R, C, E)$ where $E = \{(i,j) \in R \times C \mid a_{ij} \neq 0\}$. Let $M(\mathcal{G})$ be the number of edges in a maximum cardinality matching in the bipartite graph $\mathcal{G}$. If $|R| = |C|$, then the usual expansion of $\det A_{RC}$ has at least one non-zero term precisely when $M(\mathcal{G}_{RC}) = |R|$, and when $A$ is "general", we expect the converse of this to be true as well. This reasoning leads us to assume henceforth that $A$ has the

**Matching Property (MP):** $\operatorname{rank} A_{RC} = M(\mathcal{G}_{RC})$ for all $R$ and $C$.

For example, $A_1$ above does *not* satisfy (MP) whereas $A_2$ does. Note that if the entries of $A$ are independent algebraic indeterminates then (MP) is satisfied.

Since $T$ must be non-singular, $\mathcal{G}(T)$ has a perfect matching which we can assume without loss of generality is the main diagonal. We can further assume that $t_{ii} = 1$, $i = 1,2,\ldots,m$ by scaling the rows of $T$, so that the non-zero entries in row $i$ of $T$ indicate the multipliers for the rows to be added to row $i$ of $A$. Viewed in this way, (SP) breaks down into $m$ one row sparsity problems (ORSP$_i$), $i = 1,2,\ldots,m$. (ORSP$_i$) is the problem:

Find $\{\lambda_k,\ k \neq i\}$ so that

$$\bar{A}_{i,\bullet} = A_{i,\bullet} + \sum_{k \neq i} \lambda_k A_{k,\bullet} \tag{3}$$

has the minimum possible number of non-zeros.

Not all solutions to (3) are equally good. Since we expect that the amount of arithemetic needed to do the calculations in (3) depends upon the number of rows with non-zero multipliers, ideally we would also like to solve the Strong (ORSP$_i$):

Among all optimal solutions to (3), find one that minimises

$$|\{k \mid \lambda_k \neq 0\}|.$$

It is not clear at this point that we can solve (SP) by successively solving (ORSP$_i$) for $i = 1, 2, \ldots, m$; nevertheless we shall concentrate on (ORSP$_1$) in the remainder of this Section.

A set of multipliers $\{\lambda_k \mid k > 1\}$ for (3) when $i = 1$ defines the following index subsets:

$$\begin{aligned}
U &= \{k > 1 \mid \lambda_k \neq 0\}, \\
H &= \{j \mid a_{1j} = 0 \text{ and } a_{1j} \neq 0\}, \\
S &= \{j \mid a_{1j} = 0 \text{ and } a_{1j} = 0 \text{ and } a_{kj} \neq 0 \text{ for some } k \in U\}, \\
G &= H \cup S, \\
F &= \{j \mid a_{1j} \neq 0 \text{ and } a_{1j} = 0\}, \\
P &= F \cup S = \{j \mid a_{1j} = 0 \text{ and } a_{kj} \neq 0 \text{ for some } k \in U\}, \text{ and} \\
Z &= \{j \mid a_{1j} = 0\}.
\end{aligned}$$

That is, $U$ is the set of used rows; $H$, the set of hit columns, where a non-zero was changed to a zero; $S$, the set of saved columns, where a zero that we would have expected to be filled-in (since $a_{kj} \neq 0$) was not filled-in; $G$, the set of good columns, where the entry was actively manipulated for the better; $F$ is the set of filled-in columns; $P$ is the set of potential fill-in columns; and $Z$ is the set of zero columns. The net decrease in non-zeros in row 1 is then $|H| - |F|$, which we want to maximise to solve (ORSP$_1$). The next theorem states the intuitive result that if $k$ columns are affected for the good, then at least $k$ independent rows must have been used (we omit the technical proof).

**Theorem 2.** For any set of multipliers, $M(\mathcal{G}_{UG}) = |G|$, and hence $\operatorname{rank} A_{UG} = |G|$. □

Theorem 2 implies in particular that $|U| \geq |G|$ always holds. If $|U| > |G|$, we can select a $|G|$-subset of $U$ which perfectly matches to $G$ and use the corresponding square non-singular (by (MP)) submatrix of $A$ to zero out $A_{1G}$, thus achieving the same result with less work. Conversely, if $A_{RC}$ is a square submatrix with a perfect matching, Theorem 2 ensures that if we use $A_{RC}$ to zero out $A_{1C}$, then $G = C$, *i.e.*, only non-zeros in $C$ are hit, and fill-in occurs in

every position where it would be expected. That is, Theorem 2 shows the crucial fact that (MP) implies that there is no "unexpected cancellation."

Hence, we can assume that the canonical situation is that $|U| = |G|$ and $\mathcal{G}_{UG}$ has a perfect matching. Then $A_{UG}$ is non-singular by (MP), so the $\{\lambda_k\}$ will be uniquely determined by solving

$$\lambda^T A_{UG} = A_{1G}. \tag{4}$$

Equation (4) allows us to think of the $\{\lambda_k\}$ as coming from $U$ and $G$ rather than vice versa, thereby reducing (SP) to the more combinatorial problem of finding optimal $U$ and $G$.

Thus we need only consider all possible $U$, and for each $U$ consider only the $G$ which match perfectly into $U$. There are potentially many possible ways to select $G \subseteq \{1, 2, \ldots, n\}$ so that $G$ perfectly matches to $U$. The next theorem shows that for a given $U$ it suffices to check only one such $G$.

**Theorem 3.** Let $G_1$ and $G_2$ be two sets of columns that perfectly match into $U$, and denote the set of hit columns corresponding to $G_i$ by $H_i$, $i = 1, 2$, etc. Then

$$|H_1| - |F_1| = |H_2| - |F_2|.$$

*Proof.* Note that $P$ depends only on $U$ and not on $G_i$. Then it is easy to see that $|H_i| = |U| - |S_i|$ and $|F_i| = |P| - |S_i|$, so that $|H_i| - |F_i| = |U| - |P|$, $i = 1, 2$. □

If we fix a full-rank matching $M$ in $\mathcal{G}_{\bullet\bullet}$, then any row subset $U$ induces a unique matched column subset $G$ relative to $M$. Any such $(U, G)$ pair will have a perfect matching, namely $M$ restricted to $A_{UG}$, so (MP) ensures that $A_{UG}$ will be non-singular. Hence the multipliers can be found as in (4). Theorem 3 ensures that the best $(U, G)$ pair from among this restricted class of such pairs will solve ($\text{ORSP}_1$).

Letting the dependence of $P$ on $U$ be explicit, through (MP), Theorem 2 and Theorem 3 we have reduced the apparently algebraic problem ($\text{ORSP}_1$) into the purely combinatorial one of maximizing $|U| - |P(U)|$ over all $U \subseteq \{2, \ldots, m\}$. Define $R = \{2, \ldots, m\}$ and $\overline{U} = R \setminus U$. Then

$$\max_U \big(|U| - |P(U)|\big) = (m-1) - \min_U \big(|P(U)| + |\overline{U}|\big). \tag{5}$$

By definition of $\overline{U}$ and $P(U)$, every non-zero in $A_{RZ}$ (the **zero-section** of row 1 of $A$) is contained in either a row in $\overline{U}$ or a column in $P(U)$. If we call rows and columns **lines**, then in this situation we say that $A_{RZ}$ is **covered** by the lines in $\overline{U} \cup P(U)$. Clearly any such covering of $A_{RZ}$ by lines can be written as $\overline{U} \cup P(U)$ for some $U \subseteq R$, so by (5), finding $\max_U \big(|U| - |P(U)|\big)$ is equivalent to finding a minimum covering of $A_{RZ}$ by lines. But by the classic theorem of **König and Egervary** (see Lawler (1976) p. 190), such a minimum cover can be computed through a maximum matching in $\mathcal{G}_{RZ}$:

**Theorem 4.** $M(\mathcal{G}_{RZ}) = \min_U(|P(U)| + |\overline{U}|)$, and a maximum matching and a minimum covering by lines are dual combinatorial objects. □

By the duality theory of matching algorithms, if we find a maximum matching in $\mathcal{G}_{RZ}$ through a labelling algorithm, then an optimum $U$ for ($ORSP_1$) is the set of rows reachable via an alternating path from an unmatched row. That is, Theorem 4 shows that this $U$ solves the right-hand side of (5), so it also solves the left-hand side, and so must solve ($ORSP_1$).

Even better, the next theorem shows that the optimum $U$ defined above also solves the Strong ($ORSP_1$). For a network flow problem with source $s$ and optimal flow $f$, define the **standard** minimum cut $K^* = \{i \mid$ there is an augmenting path $s \rightarrow i$ under $f\}$. Note that the optimal $U$ defined above is a standard minimum cut for the usual way of solving a maximum cardinality bipartite matching problem by converting it to an equivalent network flow problem.

**Theorem 5.** In a given network, the standard min cut is a subset of every min cut. Thus the standard min cut is the same for every optimal flow, hence it is well-defined and has minimum cardinality among all min cuts (see Ford and Fulkerson (1962) p. 13 for a proof). □

Theorems 4 and 5 together imply that we can solve the Strong ($ORSP_1$) through maximum matching, and that the optimal $U$ is unique.

## 3. Two Algorithms for (SP)

Once we have found the optimal $U$ for each row $i$ (say, $U_i$) through matching, as noted above we can easily generate the sets $G_i$ of columns by choosing $G_i$ to be the set of columns that matches into $U_i^*$ under the fixed matching $M$. These $(U_i^*, G_i)$ pairs completely determine the non-zero off-diagonal entries of a matrix $T^*$ as defined by (4). The question arises: is $T^*$ non-singular?

To answer this question, it is necessary to investigate what the uniqueness properties of the $U_i$ imply for the structure of $T^*$. Define a directed graph $D$ with vertices $V = \{1, \ldots, m\}$ and edges $E = \{(k, i) \mid k \in U_i^*\}$; thus $D$ represents the sparsity pattern of $T^*$. If the row indices of $A$ and $T^*$ are ordered consistent with the strong component decomposition of $D$, then the decomposition induces a block lower-triangular structure on $T^*$, where the diagonal blocks of $T^*$ correspond to the strong components of $D$.

**Theorem 6.** If $l \in U_k^*$ and $k \in U_i^*$ then $l \in U_i^*$.

*Proof.* For ease of notation, let $U = U_i^*$, $\overline{U} = \{1, \ldots, m\} \setminus \{i\} \setminus U_i^*$, $P = P(U_i^*)$, and $\overline{P} = Z_i \setminus P(U_i^*)$. Thus $U$ and $\overline{U}$ partition the rows of the zero-section of row $i$ of $A$, and $P$ and $\overline{P}$ partition the columns. Recall that the rows in $\overline{U}$ and the columns in $P$ are a minimum cover of the zero-section of row $i$ of

$A$ by lines. Thus $A_{U\overline{P}} = 0$, and, since $k \in U$, $A_{k\overline{P}} = 0$. By the minimality of this cover, the submatrix $A_{U\overline{P}}$ has a row-perfect matching, and so $|\overline{U}|$ lines are necessary to cover it. Let $L^*$ be the standard minimum set of lines covering the $k^{\text{th}}$ zero-section. Since $A_{k\overline{P}} = 0$ and $k \notin \overline{U}$, the submatrix $A_{U\overline{P}}$ is part of the zero-section of row $k$ and so must be covered by the lines in $L^*$. Consider the set of lines $L = L^* \cup \overline{U} \setminus \overline{P}$. Since the only non-zeros in the columns in $\overline{P}$ of the $k^{\text{th}}$ zero-section occur in rows in $\overline{U}$, $L$ is a cover for the zero-section for row $k$. The only change in lines between $L^*$ and $L$ is in lines passing through $A_{U\overline{P}}$; since $L$ has only $|\overline{U}|$ lines passing through $A_{U\overline{P}}$, the minimum possible number, $L$ must also be a minimum cover. Finally, $L$ contains at least as many rows as $L^*$, so that the $U$ associated with $L$ has at most as many rows as the $U$ associated with $L^*$, namely $U_k^*$. But $U_k^*$ has the minimum possible number of rows for *any* minimum cover of the zero-section of row $k$, so $L = L^*$. But this implies that $U_k^* \subseteq U_i^*$. □

The conclusion of Theorem 6 is precisely that the graph $D$ is transitively closed. This implies that the blocks of the block lower-triangular partition of $T^*$ are either completely dense or all zero. In particular, $U_i^* \cup \{i\} = U_k^* \cup \{k\}$ for $i$ and $k$ in the same strong component of $D$. These observations allow us to prove the following.

**Theorem 7.** $T^*$ is non-singular.

*Proof.* Since $T^*$ is block lower-triangular, it suffices to show that the diagonal blocks of $T^*$ are non-singular. A typical diagonal block is indexed by the vertices in some strong component, say $B$. As shown above, the set $B^* = U_i^* \cup \{i\}$ is the same for all $i \in B$, and $B \subseteq B^*$. Assume for convenience that the fixed matching $M$ is such that row $i$ matches to column $i$, $i = 1, \ldots, m$. If $B = B^*$, then the diagonal block associated with $B$ is clearly just a re-scaling of $(A_{BB})^{-1}$ ($A_{BB}$ is non-singular by (MP)), and so is non-singular. Otherwise, let $L = B^* \setminus B$. Then the diagonal block associated with $B$ is a re-scaling of the bottom right corner of the matrix

$$\begin{pmatrix} A_{LL} & A_{LB} \\ A_{BL} & A_{BB} \end{pmatrix}^{-1} = \begin{pmatrix} \overline{A}_{LL} & \overline{A}_{LB} \\ \overline{A}_{BL} & \overline{A}_{BB} \end{pmatrix}$$

(this matrix is non-singular by (MP)). But it is well-known that $\overline{A}_{BB}$ is non-singular if and only if $A_{LL}$ is non-singular. But $A_{LL}$ is indeed non-singular by (MP). □

Since $T^*$ is non-singular, we can use it to transform $A$ into $\overline{A}$. This way of generating $\overline{A}$ processes each row in parallel, *i.e.* each row is solved relative to the original matrix rather than relative to a partially transformed matrix. We call this procedure the **Parallel Algorithm (PA)**.

**Theorem 7.** (PA) solves (SP) when $A$ satisfies (MP).

*Proof.* Each row of $A$ is made as sparse as possible in $\overline{A}$. □

The "parallelism" of (PA) seems unsatisfactory for two reasons. First, it is more natural to process $A$ *sequentially*, *i.e.* by solving each row's matching problem on the partially reduced $A$ whose previous rows have already been processed. Second, by processing $A$ sequentially we can overwrite $\overline{A}$ on $A$, thereby saving space, and, as we shall see later, the optimal $U$'s can only get smaller, thus also saving time in solving equations (4). More formally, consider the Sequential Algorithm (SA):

Given $A$.
For $i = 1, 2, \ldots, m$ do.
  Use matching in the $i^{\text{th}}$ zero-section of $A$ to find $U$.
  Find some $G$ so that $A_{UG}$ is non-singular.
  Replace $A_{i*}$ by $A_{i*} + \sum_{k \in U} \lambda_k A_{k*}$, where the $\lambda_k$ are defined by (4).
  End.
$A$ is the output. Stop.

We want to show that the output of (SA) solves (SP). The replacement step is equivalent to left-multiplying the current $A$ by a non-singular elementary matrix, so the output $A$ is row-equivalent to the input $A$. This also implies that we must be able to find a suitable $G$ at each iteration, or else $A$ would have lost rank at some point. Thus it remains only to show the following.

**Theorem 8.** (SA) produces the same final number of non-zeros as (PA).

*Proof.* Denote the optimal $U$ for row $i$ under (PA) by $U_i^*$, under (SA) by $U_i$, and inductively assume that $U_k \subseteq U_k^*$ for all $k < i$. Denote the original $A$ by $A^0$, the $A$ just before replacing the $i^{\text{th}}$ row by $A^i$, and the rows and columns of the $i^{\text{th}}$ zero-section of $A^0$ (which is the same for $A^i$) by $R$ and $Z$ respectively. Recall that $L^* = R \setminus U_i^* \cup P(U_i^*)$ is a minimum covering of $A_{RZ}^0$ by lines. Suppose that $k \in U_i$ for some $l \in U_i^*$, $l < i$. By the induction hypothesis $k \in U_l$ and $l < i$ imply that $k \in U_l^*$, so $k \in U_i^*$ by Theorem 6. Thus every row $l \in U_i^*$, $l < i$, that may have been changed in going from $A^0$ to $A^i$ maintains the property that it is zero in the columns in $Z \setminus P(U_i^*)$. This means that $L^*$ is also a covering of $A_{RZ}^i$ by lines. Thus

$$|L^*| = M(\mathcal{G}_{RZ}^0) = \operatorname{rank} A_{RZ}^0 = \operatorname{rank} A_{RZ}^i \leq M(\mathcal{G}_{RZ}^i) \leq |L^*|, \qquad (6)$$

where the third equality holds because $A_{RZ}^i$ is a non-singular transformation of $A_{RZ}^0$. Thus the parallel cover $L^*$ is also minimum for $A_{RZ}^i$, though it may no longer be the minimum cover with the minimum cardinality $U$. However, Theorem 5 ensures that $U_i \subseteq U_i^*$, verifying the induction.

The improvement in non-zeros in row $i$ under (SA) is $(m-1) - M(\mathcal{G}_{RZ}^i)$. But (6) shows that this is equal to $(m-1) - M(\mathcal{G}_{RZ}^0)$, which is the improvement in non-zeros in row $i$ under (PA). □

Theorem 9 together with the preceding remarks prove the final theorem.

**Theorem 10.** (SA) also solves (SP) under (MP). □

It is easy to get a good bound on the running time of the combinatorial part of both (PA) and (SA). Let $\nu$ be the number of non-zeros in $A$, which we can assume is greater than $n$. We use the following trick to reduce the running time of both (PA) and (SA). For (SA) as well as (PA), find a fixed initial maximum matching on $A$; this takes $O(m\nu)$ operations. Then, when finding a maximum matching in a zero-section, copy over the part of the fixed matching that lies in the columns of the zero-section as the starting matching; this copying takes $O(m^2)$ time. Since every initially unmatched row in the zero-section matches to some column outside the zero-section in the fixed matching, the number of unmatched rows in the starting solution for row $i$'s zero-section can be at most the number of non-zeros in row $i$. Thus the total number of augmentations needed over all rows is $O(\nu)$. Since each augmentation is an $O(\nu)$ operation, we get an $O(\nu^2)$ overall bound for the combinatorics.

The time needed to do the numerical part of (PA) and (SA) can be bounded as follows. In the worst case we will have to solve a linear system like (4) of dimension $O(m)$ for each one of $m$ rows. Solving one such system is bounded by $O(m^3)$, so the numerical part is bounded by $O(m^4)$ overall. However, we have assumed that $A$ is sparse, and there are sparse equations routines that can solve a system like (4) in time more like $O(m^2)$. In practice we expect to see only $O(1)$ rows whose linear systems are really as large as $O(m)$; most linear systems will be of size $O(1)$. Thus under favorable circumstances the numerical computations could take time as small as $O(m^2)$.

## 4. Practicalities

Very few real-life matrices satisfy (MP). In light of Theorem 5 we cannot hope to actually solve (SP) on all real matrices, but we can try to apply one of our algorithms or a variant as an "optimal" heuristic. Ideally, when we apply our "real" algorithms to real, full-rank matrices, they would be guaranteed to achieve at least the increase in sparsity that an "ideal" algorithm would achieve on a matrix with the same sparsity pattern that *did* satisfy (MP).

It is difficult to anticipate unexpected cancellation with real matrices. A parallel type of algorithm is therefore unsuitable, as it has to proceed without knowing where cancellation takes place. On the other hand, a sequential type of algorithm can take stock of the cancellation that arises at each step. However, guaranteeing performance becomes more subtle in the presence of cancellation. Consider the full-rank matrix

$$A = \begin{pmatrix} 1 & 3 & 0 & 5 & 5 \\ 2 & 1 & 4 & 0 & 0 \\ 0 & 3 & 0 & 5 & 5 \end{pmatrix}.$$

Any sequential algorithm will pick $U_1 = \{3\}$, and could pick $G_1 = \{2\}$. This transformation unexpectedly zeros out columns 4 and 5 of row 1. Thus if we naively process row 2 using this new row 1, we will choose $U_2 = \{1\}$. But the parallel $U_2^* = \emptyset$, which does not contain $U_2$ as required by the induction hypothesis of Theorem 8, so we can no longer guarantee that our final answer will be as good as the ideal. (A close reading of that proof of Theorem 8 will reveal that the only way that this difficulty can arise is when $\text{rank}\, A_{RZ} < M(\mathcal{G}_{RZ})$ for some zero-section; in the second zero-section of this example, $\text{rank}\, A_{RZ} = 1 < 2 = M(\mathcal{G}_{RZ})$.)

A simple trick will avoid this problem. As we perform (SA), we certainly know at each step where we expect the non-zeros to occur for subsequent steps. If we do encounter unexpected cancellation, we can merely pretend that there is still a non-zero in the cancelled position. That is, subsequent matchings are performed as if no unexpected cancellation ever took place, though we keep track of which "non-zeros" are really zeros. Then the proof of Theorem 8 becomes valid once again, and the modified (SA) is now guaranteed to produce an answer at least as good as the "ideal" answer.

We now make some remarks about implementing (SA). Linear constraints are usually presented as a mixture of equalities and inequalities. If these constraints are converted to the form (1) by adding a slack variable to each inequality row, it is easy to see that there is always a maximum cardinality matching in which every inequality row is matched to its slack column. It is also easy to see that such rows can never be profitably used in the optimal $U$ for any other row, since the slack variable will always unavoidably fill-in its column. (In fact, by this same reasoning, if $A$ is known to have an embedded identity matrix, then $A$ must already be optimally sparse.) Hence (SA) will still work correctly if we merely treat inequality rows as if they were *unmatched*, without having to explicitly create a slack variable at all. This phenomenon implies that (SA) will tend to find better solutions for systems with a high proportion of equality constraints.

## 5. Further Questions and Conclusion

Trying to solve the Sparsity Problem as described in this paper raises some interesting questions. From an applications point of view, the chief question is: Does (SA) help in practice or not? The answer to this question must come from empirical experience with (SA) on various problems. We have implemented a preliminary version of (SA) for this purpose; our results so far are encouraging, but we have by no means conclusively demonstrated the usefulness of (SA). We expect to report our computational experience with (SA) in the near future.

Although it is necessary to keep unexpected zeros as phantom non-zeros to guarantee the performance of (SA) on real matrices, it is certainly feasible

to run the algorithm without this artifice. Can any guarantees be made in this case? Does this make much difference in practice? Alternatively, since "lucky" cancellation has been observed in nearly all real examples we have tried, is there some efficient heuristic for (SP) that can take advantage of this and outperform (SA), perhaps restricted to some subset of interesting problems with special structure? Finally, what happens when we try to apply these algorithms to rank-deficient matrices?

We shall continue our research on (SP) and shall try to answer some of these questions in future papers.

## Appendix.

*Proof.* (of Theorem 1) This Theorem and its proof are due to L. Stockmeyer (1982). See Garey and Johnson (1979) for the definitions of the concepts used in this proof.

The problem that we shall reduce to (SP) is

**Simple Max Cut**: Given an undirected graph $\mathcal{G} = (V, E)$, partition the nodes of $\mathcal{G}$ into $P$ and $V \setminus P$ so as to maximize

$$|\{\{i,j\} \in E \mid i \in P, j \in V \setminus P\}|.$$

Let $n = |V|$, $m = |E|$, let $A(\mathcal{G})$ be the usual (0,1) node-arc incidence matrix of $\mathcal{G}$, and let $A_i$ be the $n \times 2m$ matrix which is all zero except for row $i$, which is half $+1$ and half $-1$. Let $e$ be the $2m$-vector of all ones and let $f$ be the $(2m(n+1)+1)$-vector of ones. Now suppose we could solve (SP) on the matrix

$$B(\mathcal{G}) = \begin{pmatrix} 0 & e & e & \cdots & e & f \\ A(\mathcal{G}) & A_1 & A_2 & \cdots & A_n & 0 \end{pmatrix}.$$

Clearly some row of the optimal $T$ for $B(\mathcal{G})$, which we may assume without loss of generality is the first, will be a multiple of $(1, \epsilon_1, \epsilon_2, \ldots, \epsilon_n)$ where $\epsilon_i = \pm 1$ for all $i \in V$. (Because of the size of $f$ the first column of $T$ must be $(1, 0, 0, \ldots, 0)$ and so this choice for the first row of $T$ causes no singularity problems.) Let $P = \{i \mid \epsilon_i = +1\}$. Then the number of non-zeros in the first row of $\overline{B(\mathcal{G})}$ is clearly

$$(2m(n+1)+1) + n^2 + (m - |\{\{i,j\} \in E \mid i \in P, j \in V \setminus P\}|). \qquad (7)$$

But since (7) is minimized by the optimal $T$, $P$ also solves the Simple Max Cut Problem for $\mathcal{G}$. □

## References.

Bunch, J. R. and D. J. Rose, eds., "Sparse Matrix Computations," Academic Press (New York, 1976).

Duff, I. S., ed., "Sparse Matrices and their Uses," Academic Press (New York, 1980).

Ford, L. R. and D. R. Fulkerson, "Flows in Networks," Princeton University Press (Princeton, 1962).

Garey, M. L. and D. S. Johnson, "Computers and Intractibility," Freeman (San Francisco, 1979).

Lawler, E. L., "Combinatorial Optimization," Holt, Rinehart and Winston (New York 1976).

Papadimitriou, C. H. and K. Stieglitz, "Combinatorial Optimization: Algorithms and Complexity," Prentice Hall (Englewood Cliffs, 1982).

Stockmeyer, L. J., personal communication (1982).

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|
| 1. REPORT NUMBER: SOL 82-13 | 2. GOVT ACCESSION NO.: AD-A120 902 |
| | 3. RECIPIENT'S CATALOG NUMBER: |
| 4. TITLE (and Subtitle): A Fast Algorithm That Makes Matrices Optimally Sparse | 5. TYPE OF REPORT & PERIOD COVERED: Technical Report |
| | 6. PERFORMING ORG. REPORT NUMBER: |
| 7. AUTHOR(s): Alan J. Hoffman; S. Thomas McCormick | 8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0267; DAAG29-79-C-0110 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: Department of Operations Research - SOL, Stanford University, Stanford, CA 94305 | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR-047-143 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research - Dept. of the Navy, 800 N. Quincy Street, Arlington, VA 22217 | 12. REPORT DATE: September 1982 |
| | 13. NUMBER OF PAGES: 11 pp. |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) | 15. SECURITY CLASS. (of this report): UNCLASSIFIED |
| | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: |

16. DISTRIBUTION STATEMENT (of this Report)

This document has been approved for public release and sale; its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

sparse matrices
bipartite matching
linear constraints

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

SEE REVERSE SIDE

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

**SOL 82-13 A FAST ALGORITHM THAT MAKES MATRICES OPTIMALLY SPARSE, Alan J. Hoffman, S. Thomas McCormick, (September 1982, 11 pp.)**

Under a non-degeneracy assumption on the non-zero entries of a given sparse matrix, a polynomially-bounded algorithm is presented that performs row operations on the given matrix which reduce it to a sparsest possible matrix with the same row space. For each row of the matrix, the algorithm performs a maximum cardinality matching on the bipartite graph associated with a submatrix which is induced by that row. The dual of the optimal matching then specifies the row operations that will be performed on that row. We also describe a variant algorithm that processes the matrix in place, thus conserving storage and time. The modifications needed to apply the algorithm to matrices that do not necessarily satisfy the non-degeneracy assumption are also described. A particularly promising application of this algorithm is in the reduction of linear constraint matrices.